```
? ds

Set      Items   Description
S1       37130   (SPIDER OR WANDER OR CRAWL??? OR SCOOTER)(S)(WEB OR INTERN-
             ET)
S2      257766   (ELIMINAT????? OR AVOID???? OR PREVENT?????)(7N)(DUPLICAT?-
            ???? OR IDENTICAL OR SAME)
S3          57   S1(S)S2
S4           4   S3(S)(BUFFER? ? OR FIFO? ? OR CACHE? ? OR DISK OR DISC)
S5          53   S3 NOT S4
S6           8   S5(S)(URL OR ADDRESS? ? OR FINGERPRINT? ?)
S7         385   S1(S)(BUFFER? ? OR QUEUE? ? OR FIFO? ? OR CACHE? ?)
S8          39   S7(S)(DISK OR DISC)
S9          39   S8 NOT S4
S10         38   S9 NOT S6
S11          1   S9 AND S6
S12         38   S10 NOT S11
?
```

Dialog search

05-09-2005

Freeform Search

Database: US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Term: L13 same (disk or disc)

Display: 10 **Documents in Display Format:** - **Starting with Number** 1

Generate: ○ **Hit List** ◉ **Hit Count** ○ **Side by Side** ○ **Image**

Search Clear Interrupt

Search History

DATE: Monday, May 09, 2005 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=ADJ			
L14	L13 same (disk or disc)	9	L14
L13	L12 same (url or address$2 or fingerprint)	51	L13
L12	l1 same (buffer or queue or fifo)	86	L12
L11	l1 same L10	4	L11
L10	(buffer or cache or fifo or queue) near3 (full or fill$3)	35825	L10
L9	l1 same L8	2	L9
L8	(address$2 or url) near2 filter$3	3932	L8
L7	l1 same fingerprint	22	L7
L6	L5 not l3	69	L6
L5	l1 same L4	81	L5
L4	(eliminat$4 or avoid$4 or prevent$4) near7 (duplicat$4 or replicat$4 or identical$2 or sam$2)	281237	L4
L3	l1 same L2	31	L3
L2	(duplicat$4 or replicat$4 or identical or sam$2) near2 (address$2 or url)	50335	L2
L1	(spider or wander or crawl$3) same (web or internet)	7395	L1

Links in a web page may refer to web pages that are stored in the same or different host computers.

A web crawler is a program that automatically finds and downloads documents from host computers in an Intranet or the world wide web. A computer with a web crawler installed on it may also be referred to as a web crawler. When a web crawler is given a set of starting URL's, the web crawler downloads the corresponding documents. The web crawler then extracts any URL's contained in those downloaded documents. Before the web crawler downloads the documents associated with the newly discovered URL's, the web crawler needs to find out whether these documents have already been downloaded. If the documents associated with the newly discovered URL's have not been downloaded, the web crawler downloads the documents and extracts any URL's contained in them. This process repeats indefinitely or until a predetermined stop condition occurs.

Typically, to find out whether the documents associated with a set of discovered URL's have already been downloaded or are scheduled to be downloaded, the web crawler checks a directory of document addresses. These document addresses are URL's that correspond to documents which have either already been downloaded or are scheduled to be downloaded; for convenience, these documents will be referred to as downloaded documents. The directory stores the URL's of the downloaded documents, or representations of the URL's. The set of URL's in downloaded documents could potentially contain addresses of every document on the world wide web. As of 1999 there were approximately 800 million web pages on the world wide web and the number is continuously growing. Even Intranets can store millions of web pages. Thus, web crawlers need efficient data structures to keep track of downloaded documents and any discovered addresses of documents to be downloaded. Such data structures are needed to facilitate fast data checking and to avoid downloading a document multiple times.

Typically, the set of downloaded document addresses is stored in disk storage, which has relatively slow access time. One example of a method designed to facilitate fast data checking and to avoid downloading a document multiple times is disclosed in U.S. Patent Application Serial No. 09/433,008, filed November 2, 1999. That document discloses storing address representations on disk, and using an efficient address representation to facilitate fast

MMM 5-9-2005

now U.S. Pat. 6,301,614

consuming operation. Therefore buffer B 107 is typically made fairly large so as to minimize the frequency of such merge operations.

During the merge process, which is an ordered merge, fingerprint N_k must be inserted in the fingerprint disk file 113 in the proper location, as illustrated in Figure 6, so that the disk file 113 remains ordered. This requires the disk file to be completely re-written. To avoid this lengthy rewrite process, in a preferred embodiment, the fingerprint disk file may be sparsely-filled, using open addressing. For this embodiment, the fingerprint disk file represents a hash table, with a substantial proportion of the table, for example 50% or 75%, being empty entries, or "holes."

In this embodiment, in order to determine whether a particular fingerprint N_k is in the disk file, the hash of the fingerprint is computed. In one embodiment, only a prefix of the fingerprint is used for the hash value. The hash value is the starting position for searching through the fingerprint disk file. The disk file is searched sequentially, starting at the starting position, for either a match or a hole. If a hole is found, the fingerprint N_k is stored in that hole; if a match is found, N_k is discarded. Thus, there is only one write to the disk file for each fingerprint not already present in the disk file, and the size of the disk file is not a factor in the merge time. When the disk file becomes too full – for example, when only 25% of the slots in the disk file are holes – the file must be completely rewritten into a new, larger file. For example, the new file may be doubled in size, in which case the amortized cost of maintaining the file is constant per fingerprint in the hash table. It will be appreciated that the use of open addressing a sparsely-filled disk file drastically reduces the disk re-writing required during a merge.

In one embodiment, the disk file may be divided into sparse sub-files, with open-addressing used for each sub-file. An index may be used to identify the range of fingerprint hash values located in each sub-file, or an additional hash table may be used to map fingerprints to the various sub-files. When a sub-file becomes too full, it may be re-written into a new, larger file, but the entire disk file need not be re-written.

In another aspect of the present invention, an efficient addressing scheme may be used for either a sparse disk file, or a disk file consisting of a set of sparse sub-files. In this addressing scheme, discussed in U.S. patent application 09/433,008, filed November 2, 1999 (now US Pat. 6,301,614 hereby incorporated by reference in its entirety), each fingerprint is composed of two

MMM 5/9/2005